A laptop was stolen from Grad division with names, SS #, birth dates of almost $10^6$ former grad students at UC Berkeley. The thief may not know what they have! Sensitive data allowed on portables? Good idea...NOT!

98,389 UC Identity thefts?! ⇒ newscenter.berkeley.edu/security/grad/
How to Design a Processor: step-by-step

1. Analyze instruction set architecture (ISA) => datapath requirements
   - meaning of each instruction is given by the register transfers
   - datapath must include storage element for ISA registers
   - datapath must support each register transfer

2. Select set of datapath components and establish clocking methodology

3. Assemble datapath meeting requirements

4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.

5. Assemble the control logic (hard part!)
Step 3: Assemble DataPath meeting requirements

• Register Transfer **Requirements** ⇒ Datapath **Assembly**

• Instruction Fetch

• Read Operands and Execute Operation
3a: Overview of the Instruction Fetch Unit

- The common RTL operations
  - Fetch the Instruction: mem[PC]
  - Update the program counter:
    - Sequential Code: PC = PC + 4
    - Branch and Jump: PC = “something else”
3b: Add & Subtract

• \( R[rd] = R[rs] \text{ op } R[rt] \)  
  Ex.: \( \text{addU } rd, rs, rt \)

• Ra, Rb, and Rw come from instruction’s Rs, Rt, and Rd fields

\[
\begin{array}{ccccccccc}
\text{op} & \text{rs} & \text{rt} & \text{rd} & \text{shamt} & \text{funct} \\
6 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 5 \text{ bits} & 6 \text{ bits}
\end{array}
\]

• ALUctr and RegWr: control logic after decoding the instruction

• We’ve already defined register file, ALU
Clocking Methodology

- Storage elements clocked by same edge
- Being physical devices, flip-flops (FF) and combinational logic have some delays
  - Gates: delay from input change to output change
  - Signals at FF D input must be stable before active clock edge to allow signal to travel within the FF, and we have the usual clock-to-Q delay
- “Critical path” (longest path through logic) determines length of clock period
Register-Register Timing: One complete cycle

Clk

PC	Old Value	New Value

Rs, Rt, Rd,
Op, Func	Old Value	New Value

Instruction Memory Access Time

Delay through Control Logic

ALUctr	Old Value	New Value

Register File Access Time

RegWr	Old Value	New Value

ALU Delay

busA, B	Old Value	New Value

Register Write Occurs Here
3c: Logical Operations with Immediate

- \( R[rt] = R[rs] \text{ op } \text{ZeroExt}[\text{imm16}] \)

What about \( Rt \) register read??

Already defined 32-bit MUX; Zero Ext?
3d: Load Operations

- \( R_{rt} = Mem[R_{rs} + \text{SignExt}[\text{imm16}]] \)

Example: \( lw \ rt, rs, \text{imm16} \)
3e: Store Operations

  Ex.: sw rt, rs, imm16

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

Ex: sw rt, rs, imm16
3f: The Branch Instruction

- `beq rs, rt, imm16`
  - `mem[PC]` Fetch the instruction from memory
  - `Equal = R[rs] == R[rt]` Calculate branch condition
  - if (Equal) Calculate the next instruction’s address
    - `PC = PC + 4 + (SignExt(imm16) x 4)`
  else
    - `PC = PC + 4`
Datapath for Branch Operations

- **beq rs, rt, imm16**
  
  Datapath generates condition (equal)

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>

- Already MUX, adder, sign extend, zero
Putting it All Together: A Single Cycle Datapath

Diagram showing the components of a single cycle CPU datapath:

- Instruction Memory
- Registers
- Adder
- ALU
- Memory
- Muxes
- Control signals

Key components include:
- Instruction<31:0>
- Rs, Rt, Rd, Imm16
- RegDst, RegWr
- ALUctrl, MemWr, MemtoReg
- PC, nPC sel
- ExtOp, ALUSrc

The diagram illustrates the flow of data and control signals through the various components of the datapath.
An Abstract View of the Implementation

Datapath

Control

Ideal Instruction Memory

Instruction Address

Next Address

PC

Clk

Rw Ra Rb
32 32-bit Registers

ALU

A

B

Data Memory

Data Address

Data In

Data Out

Clk

Ideal Data Memory

Control Signals

Conditions

Instruction

Rd Rs Rt
5 5 5

Next Address

32

32

32
A. Our ALU is a synchronous device

B. We could have used tri-state devices instead of a MUX to feed busW, the register write data line

C. The ALU is inactive for memory reads or writes.
Summary: Single cycle datapath

° 5 steps to design a processor
  • 1. Analyze instruction set => datapath requirements
  • 2. Select set of datapath components & establish clock methodology
  • 3. Assemble datapath meeting the requirements
  • 4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  • 5. Assemble the control logic

° Control is the hard part
° Next time!